

Teddy: Efficient Large-Scale Dataset Distillation via Taylor-Approximated Matching

Ruonan Yu[Ⓜ], Songhua Liu[Ⓜ], Jingwen Ye[Ⓜ], and Xinchao Wang^{Ⓜ*}

National University of Singapore

{ruonan, songhua.liu}@u.nus.edu, {jingweny, xinchao}@nus.edu.sg

Abstract. Dataset distillation or condensation refers to compressing a large-scale dataset into a much smaller one, enabling models trained on this synthetic dataset to generalize effectively on real data. Tackling this challenge, as defined, relies on a bi-level optimization algorithm: a novel model is trained in each iteration within a nested loop, with gradients propagated through an unrolled computation graph. However, this approach incurs high memory and time complexity, posing difficulties in scaling up to large datasets such as ImageNet. Addressing these concerns, this paper introduces **Teddy**, a **Taylor-approximated dataset distillation** framework designed to handle large-scale dataset and enhance efficiency. On the one hand, backed up by theoretical analysis, we propose a memory-efficient approximation derived from Taylor expansion, which transforms the original form dependent on multi-step gradients to a **first-order** one. On the other hand, rather than repeatedly training a novel model in each iteration, we unveil that employing a pre-cached pool of **weak** models, which can be generated from a **single** base model, enhances both time efficiency and performance concurrently, particularly when dealing with large-scale datasets. Extensive experiments demonstrate that the proposed Teddy attains state-of-the-art efficiency and performance on the Tiny-ImageNet and original-sized ImageNet-1K dataset, notably surpassing prior methods by up to 12.8%, while reducing 46.6% runtime. Our code will be available at <https://github.com/Lexie-YU/Teddy>.

Keywords: Dataset distillation · Taylor approximation · Efficient training

1 Introduction

Deep learning has made remarkable strides in various fields, largely owing to the utilization of extensive training data [7, 9, 16, 28, 35]. However, there are associated expenses; achieving optimal performance necessitates substantial computational resources and a significant amount of GPU time, particularly for the latest state-of-the-art models [2, 9, 24, 26]. Concerning these issues, a promising and emerging field, dataset distillation (DD), also known as dataset condensation (DC), has

* Corresponding author.

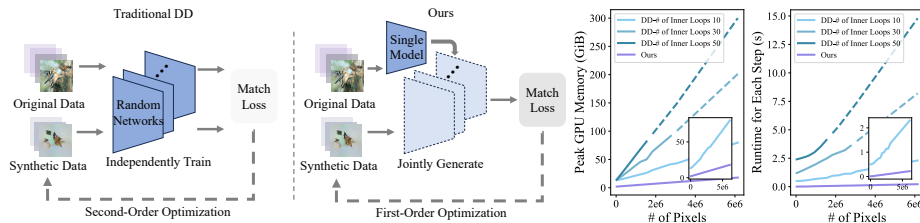


Fig. 1: Illustration of meta-learning-based methods, our method (left), and the comparison of memory and time efficiency (right). Our proposed method exhibits surprising memory and time efficiency.

recently garnered significant attention. DD concentrates on condensing the extensive original dataset into a much smaller version while retaining the entirety of the dataset knowledge so that models trained on the synthetic dataset can generalize well on real data.

Following the definition of DD, Wang *et al.* [33] in seminal propose a principled meta-learning-based algorithm: in each iteration, a novel model is trained on the synthetic dataset in a nested loop, and the trained model is meta-tested on original data, and the gradient is backpropagated through an unrolled computation graph to update current synthetic data. Although effective, such bi-level optimization results in high memory and time complexity in practice. Specifically, on the one hand, each update of the current synthetic dataset necessitates an iterative meta-train process from a new model, making the time required for each step notably prolonged. On the other hand, caching the whole computation graph for back-propagation demands significant GPU memory. As illustrated in Fig. 1, for instance, under the setting of a 10-class dataset with 50 images per class, each update requires over 50 GiB GPU memory, and the entire data generation takes hundreds of hours, let alone when condensing the large-scale ImageNet dataset, which has thousands of times optimization pixel space of the small dataset like CIFAR10.

To address the challenges above, subsequent research endeavors have focused on devising surrogate objectives to mitigate the bi-level nature of DD. While experiments have demonstrated their enhanced efficiency and effectiveness, these approaches still necessitate extensive forward and backward passes in neural networks for computation. For instance, methods such as gradient matching [39, 41] entail sampling gradients of current synthetic datasets and original data across a multitude of networks. Similarly, distribution matching [31, 40] involves computing feature statistics within numerous networks. Additionally, training-trajectory-matching techniques [3, 5] necessitate the preparation of hundreds of training trajectories containing thousands of model checkpoints altogether. The substantial computational demands of current algorithms pose significant challenges when scaling up to handle large-scale datasets.

In light of the analysis presented above, we rethink and deeply explore the primary DD optimization objectives in this paper. Concentrating on the main

bi-level optimization problem, we propose an efficient approximation solution derived from Taylor expansion to the original one, transforming the original paradigm dependent on multi-step gradients or the second-order optimization to a first-order one. Moreover, we further simplify this Taylor approximation solution by pre-caching a pool of weak models, which enhances time efficiency and has also been verified as a more advantageous alternative to training models from scratch or using fully converged ones. Specifically, two practical cases are taken into consideration here to obtain weak teachers: prior and post-generation. Both efficiently generate the model pool from a single base model.

To demonstrate the rationality of our proposed method, we provide a comprehensive theoretical analysis. This analysis also reveals the relationships between the mainstream optimization objectives adopted by existing methods and builds a unified theoretical framework of DD. Extensive experiments also validate the superiority of the proposed Teddy. It outperforms previous state-of-the-art methods by a large margin, especially for the original-sized ImageNet-1K dataset. Our contributions can be summarized below:

- We propose an efficient large-scale dataset distillation method termed Teddy, a Taylor-approximated version of the solution to the DD. Our proposed method decouples the bi-level optimization and avoids model training within inner loops for each iteration, which can better handle large-scale datasets.
- We provide a comprehensive theoretical analysis for our proposed method, and we are the first to give a theoretical analysis linking the existing mainstream dataset distillation objectives used in drastically different dataset distillation methods. Our study establishes a unified theoretical framework for dataset distillation.
- We pre-cache weak models to decrease the time complexity introduced by training student models in nested loops for each iteration. Specifically, we consider two practical cases, the prior and post model pool generation, both efficiently generating the model pool from a single base model.
- We conduct comprehensive experiments on our proposed method, which show state-of-the-art performance. Notably, on the full-sized ImageNet-1K dataset, our proposed method outperforms previous methods by a substantial margin, reaching up to 12.8%.

2 Related Works

Dataset distillation (DD), first proposed by Wang *et al.* [33], aims to condense the large-scale dataset into a small synthetic dataset and, when performed as the training data, preserve the performance of the trained models. DD addresses the tremendous computational cost issues and improves the efficiency of downstream tasks. In recent years of rapid development, increasing research has emerged in the field of DD. Based on the variations in the optimization objectives, it can be broadly categorized into three classes: performance matching [19, 22, 33, 43], parameter matching [3, 5, 10, 20, 34, 39, 41], and distribution matching [31, 40, 42]. Moreover, some research is dedicated to improving the efficiency of data

storage space utilization and expanding the learnable information space, such as label distillation [1, 29], dataset parameterization [8, 15, 17, 18, 39], and model augmentation [38], further enhancing the distilled performance.

However, previous works in DD still have problems scaling up, including generating larger synthetic datasets, adapting to more extensive training networks, and handling more complex and massive original datasets [4, 37]. One significant reason for this issue is the bilevel optimization process, where each synthetic dataset update requires backpropagation through the unrolled graph, making it high-memory demanding. For this issue, one intuitive idea is to decouple the bilevel optimization process and unbind the network and the synthetic dataset [37], such as the practice of DM [40] and IDM [42], treating the network as the feature extractor without updating it or updating using the original dataset, and matching the feature distribution of the original and the synthetic datasets. However, there continues to be a substantial performance disparity with the performance and parameter matching methods, and recalculating the mean of the original dataset features for every iteration is an additional overhead.

A recent work, proposed by Yin *et al.* [36], utilizes the statistic information running mean and running variance of the original training datasets stored in the batch normalization layers of the well-trained model. They only adopt one single model, which may lack a comprehensive view of the entire dataset and lead to the mode collapse problem. Moreover, statistics information stored in batch normalization layers of well-trained models may overemphasize long-tail samples, which is challenging for the synthetic dataset to fit and be optimized.

3 Methodology

This section provides a comprehensive introduction to the proposed method, Teddy. We will begin by elucidating the underlying motivation, followed by an exhaustive presentation of our algorithm and theoretical analysis.

3.1 Preliminary

Given the original dataset, denoted as $\mathcal{T} = (X_t, Y_t)$, where $X_t \in \mathbb{R}^{N_t \times d}$ and $Y_t \in \mathbb{R}^{N_t \times c}$, the primary goal of DD is to generate a synthetic dataset $\mathcal{S} = (X_s, Y_s)$, where $X_s \in \mathbb{R}^{N_s \times d}$ and $Y_s \in \mathbb{R}^{N_s \times c}$, such that the model training on it can obtain a comparable performance with the original dataset. Here, N_t and N_s are the number of samples in \mathcal{T} and \mathcal{S} , respectively, and $N_s \ll N_t$. d is the number of features for each sample. c is the number of classes for classification tasks. The objective can be formulated as:

$$\begin{aligned} \mathcal{S} &= \arg \min_{\mathcal{S}} \mathbb{E}_{\theta^{(0)} \sim \Theta} [l(\mathcal{T}; \theta_S^{(T)})], \\ \theta_S^{(t)} &= \theta_S^{(t-1)} - \alpha \nabla_{\theta_S^{(t-1)}} l(\mathcal{S}; \theta_S^{(t-1)}), \end{aligned} \tag{1}$$

where Θ is the distribution of model initialization, T is the number of epochs the model trained on the synthetic dataset, $1 \leq t \leq T$, α is the learning rate, and $l(\cdot; \theta)$ is the cross-entropy loss for classification tasks.

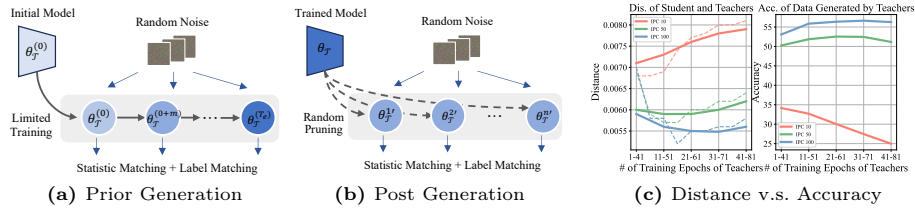


Fig. 2: (a)(b) Illustration of our proposed method. Firstly, we generate the model pool by prior or post-generation. Then, update the initialized distilled data via statistic and label matching. Lastly, input the generated distilled data with augmentation into the model pool again to obtain the soft label. (c) The left figure shows the distance between student and teacher models at different stages. The dotted line represents the distance between a single teacher and a student. The solid line represents the average of the distances within the range. The right figure shows the performance of distilled data generated from teacher models at different stages.

As Eq. 1 shows, the objective involves a bi-level optimization process. Each step of updating the synthetic dataset needs to backpropagate through an unrolled computation graph, making it inefficient for both memory and time. Moreover, it should train a novel model in a nested loop for every iteration, resulting in notable time consumption, particularly when handling complex datasets.

3.2 Taylor-Approximated Matching

In this section, we start with the fundamental solution to DD, the meta-learning-based method, progressively delving into our proposed method, Teddy, with theoretical analysis. The DD formulation is shown in Eq. 1. Practically, to enhance the generalization ability of the distilled data on the real data, the inner loop is typically configured with multiple steps, *e.g.*, $T > 10$. Considering the synthetic dataset is of small size, the model is highly susceptible to overfitting on it as training for T epochs. Therefore, it has $l(\mathcal{S}; \theta_S^{(T)}) < \epsilon$, where ϵ is close to zero. Employing the Karush-Kuhn-Tucker (KKT) conditions, the optimization objective in Eq. 1 can be transformed into the following:

$$\begin{aligned} \mathcal{S} &= \arg \min_{\mathcal{S}} [l(\mathcal{T}; \theta_S^{(T)}) + u(l(\mathcal{S}; \theta_S^{(T)}) - \epsilon)], \\ \theta_S^{(t)} &= \theta^{(0)} - \alpha \sum_{i=0}^{t-1} g_S^{(i)} = \theta_S^{(t-1)} - \alpha g_S^{(t-1)}, \end{aligned} \quad (2)$$

where $\theta^{(0)} \sim \Theta$, $g_S^{(t)} = \nabla_{\theta^{(t)}} l(\mathcal{S}; \theta_S^{(t)})$, and u is the Lagrange multiplier.

Proposition 1 *The meta-learning-based optimization objective can be Taylor-approximated as the sum of the gradient matching of the distilled data and the original data for all steps along the training trajectory of the student model.*

Proof. Here, we recursively apply the first-order Taylor expansion to unfold the unrolled computational graph. The first term of the optimization objective can be transformed into follows:

$$\begin{aligned} l(\mathcal{T}; \theta_S^{(T)}) &= l(\mathcal{T}; \theta_S^{(T-1)} - \alpha g_S^{(T-1)}) \approx l(\mathcal{T}; \theta_S^{(T-1)}) - \alpha g_{\mathcal{T}}^{(T-1)} \cdot g_S^{(T-1)} \\ &\approx l(\mathcal{T}; \theta_S^{(T-i)}) - \alpha \sum_{t=T-i}^{T-1} g_{\mathcal{T}}^{(t)} \cdot g_S^{(t)} \approx l(\mathcal{T}; \theta^{(0)}) - \alpha \sum_{t=0}^{T-1} g_{\mathcal{T}}^{(t)} \cdot g_S^{(t)}, \end{aligned} \quad (3)$$

where $\theta^{(0)} \sim \Theta$, and $g_{\mathcal{T}}^{(t)} = \nabla_{\theta^{(t)}} l(\mathcal{T}; \theta_S^{(t)})$. The first term $l(\mathcal{T}; \theta^{(0)})$ is the loss of the original dataset on the initial model $\theta^{(0)}$, and irrelevant to the optimization problem *w.r.t* \mathcal{S} . Concluded from Eq. 3, the meta-learning-based optimization objective can be transformed into the sum of the gradient matching for all steps pathing the student model training trajectory.

However, updating the distilled data should compute and accumulate the gradient matching loss at each step within the inner loop. It involves a second-order optimization problem, which may cause high computational complexity. To tackle this issue, we transfer the optimization objective into a first order one.

Proposition 2 *Gradient matching is equivalent to first-order and second-order statistic information matching in feature space if the dataset classes are balanced.*

Proof. For simplicity, we assume the case in FRePo [43], where only the last linear layer of the network (θ) is updated for synthetic data, denoted as W , while the preceding layers (f_{θ}) serve as the feature extractor. In this case, the gradient can form as follows:

$$\begin{aligned} g &= \frac{\partial l(\cdot; \theta)}{\partial W} = \frac{1}{|X|} f_{\theta}(X)^T (f_{\theta}(X)W - Y) \\ &= \frac{1}{|X|} f_{\theta}(X)^T f_{\theta}(X)W - \frac{1}{|X|} f_{\theta}(X)^T Y \\ &\stackrel{\theta^{(0)} \sim \Theta}{=} \frac{1}{|X|} \sigma^2(f_{\theta}(X))W - \frac{1}{|X|} \mu(f_{\theta}(X)). \end{aligned} \quad (4)$$

The first term of Eq. 4 is the weighted correlation of the extracted features of the dataset, and the second term is the class-wise mean of the extracted features. Under the condition of $\theta^{(0)} \sim \Theta$, the constraining covariance is essentially equivalent with variance. Also, when classes in the dataset are balanced, the class-wise mean can be replaced by global mean. For detailed proofs, please refer to the supplementary materials.

Therefore, the optimization objective in Eq. 2 can be transferred to:

$$\begin{aligned} l(\mathcal{T}; \theta_S^{(T)}) &\approx \mathbb{E}_{\theta^{(0)} \sim \Theta} \left[\sum_{t=0}^{T-1} \left(\sum_l \|\mu_l(f_{\theta_S^{(t)}}(X_s)) - \mu_l(f_{\theta_S^{(t)}}(X_t))\|_2 \right. \right. \\ &\quad \left. \left. + \sum_l \|\sigma_l^2(f_{\theta_S^{(t)}}(X_s)) - \sigma_l^2(f_{\theta_S^{(t)}}(X_t))\|_2 \right) + u \cdot l(\mathcal{S}; \theta_S^{(T)}) \right], \end{aligned} \quad (5)$$

where μ_l and σ_l^2 refer to the mean and variance of the l^{th} layer features. Here, we solve the core issue of bi-level optimization by stripping the inner loop training process from the backpropagation computational graph. That is, we match the statistical information of the original and synthetic dataset in the feature space. Specifically, we train a single student trajectory for every iteration from scratch. For each checkpoint along the trajectory, we perform the statistic information (the mean and variance in each layer) matching of the original and synthetic datasets, which largely improves the efficiency.

3.3 Model Pool Generation

Considering that we have already improved the efficiency by Eq. 5, it is still time demanding since we need to re-train the new model for each synthetic data updating. It will become significantly pronounced when handling the large-scale datasets.

To solve this problem and improve the efficiency, we should deal with the long student training trajectory and retraining student model for each iteration. We notice that for each synthetic data updating iteration, any segments of the student model training trajectory that begin with t and end with $t + m$ have:

$$l(\mathcal{T}; \theta_S^{(t+m)}) = l(\mathcal{T}; \theta_S^{(t)} - \alpha \sum_{i=t}^{t+m-1} g_S^{(i)}) \approx l(\mathcal{T}; \theta_S^{(t)} - \alpha (\sum_{i=t}^{t+m-1} g_S^{(i)}) \cdot g_{\mathcal{T}}^{(t)}), \quad (6)$$

of which the $(\sum_{i=t}^{t+m-1} g_S^{(i)}) \cdot g_{\mathcal{T}}^{(t)}$ shows that multi-step student model updating $\sum_{i=t}^{t+m-1} g_S^{(i)}$ is comparable with single-step teacher updating $g_{\mathcal{T}}^{(t)}$ for appropriate interval m . Thus, each student trajectory segmentation can be replaced by a single **comparable** teacher model. The “comparable” here means the teachers have the same-level performance as students (trained on the synthetic dataset), which does not reach convergence on the real data and is **weak** in performance. Also, the distance between the students and teachers is close.

Insight from Eq. 6, to further improve the time efficiency, we construct the weak teacher model pool from two practical cases, the prior and post-generation. These strategies are both efficiently generating model pool from a single base model, as shown in Fig. 2a and Fig. 2b. Here, as we utilize weak teachers instead of fully-convergence ones, we can just use a single model as the base to generate diverse weak teachers. This strategy does not compromise performance. More specifically, if the base model is:

- at the early stage or randomly initialized, we adopt the prior-generation strategy. We pre-cache the weak teachers from the early stage of the training trajectory for every m steps, substituting the whole student training trajectory.
- at the late stage or well-trained, we propose an alternative pruning-based post-generation model pool method. Here, we employ the DepGraph [11] with random strategy, complemented by fine-tuning with very limited (*e.g.*, 0-2) steps.

Additionally, when dealing with extensive models and datasets, training weak teachers, even with a limited number of epochs, still remains a significant overhead. In this case, post-generation allows for the rapid generation of many weak models with low computational costs. Also, pruned models exhibit enhanced inference speeds, further boosting the efficiency of our method.

To verify the rationality of this strategy, we conduct validation experiments. We measure the distance between the student and teacher models at different stages and the accuracy of the distilled data generated from teacher models at different stages. Here, we use KL divergence to measure model distance, capturing the information loss when approximating one probability distribution with another. The results shown in Fig. 2c reveal a strong inverse correlation between model distance and distilled data performance. Closer distances indicate better approximation and superior performance.

Here, as we replace the student models with weak teachers, we can utilize the running mean and running variance for the original dataset from the batch normalization layers in the weak teachers instead of recalculating multiple times for each iteration. It will reduce a lot of computational costs, especially for large-scale datasets. Additionally, the classifier prior $l(\mathcal{S}; \theta_{\mathcal{S}}^{(T)})$ can also be replaced by ensemble classifier prior, as weak teachers may mislead generation direction. It will also benefit better alignment with the target classes. In summary, the final optimization objective is as follows:

$$\begin{aligned} & \sum_{t=T_b}^{T_e} \left(\sum_l \|\mu_l(f_{\theta_{\mathcal{T}}^{(t)}}(X_s)) - RM_{\theta_{\mathcal{T}}^{(t)}}^l(X_t)\|_2 \right. \\ & \left. + \sum_l \|\sigma_l^2(f_{\theta_{\mathcal{T}}^{(t)}}(X_s)) - RV_{\theta_{\mathcal{T}}^{(t)}}^l(X_t)\|_2 + u \cdot l(\mathcal{S}; \theta_{\mathcal{T}}^{(t)}) \right), \end{aligned} \quad (7)$$

where RM and RV refer to the running mean and variance stored in the batch normalization layers. T_b and T_e (both at the early stage) are the starting and end points of the teacher training trajectory for statistic information matching. We only consider performing the matching for every m steps from T_b to T_e . For more theoretical details, please refer to the supplementary materials.

3.4 Algorithm Summary

In summary, we propose an efficient large-scale dataset distillation method, Teddy, solving the core bi-level optimization problem and further improving the efficiency via Taylor approximation. The framework of Teddy is shown in Algorithm 1. We first generate model pool \mathcal{M} from single base model θ_{base} . According to the state of the θ_{base} , we choose from two efficient model pool generation strategies, prior and post-generation. Then, we follow our proposed optimization objective, Eq. 7, to update the synthetic dataset, which only requires the first-order optimization process. Here, considering the weak teacher models may mislead the data generation direction, we generate soft labels via weak teacher ensemble, which provides more richer label information expression.

Algorithm 1: Teddy Framework

Input: Original dataset \mathcal{T} , single base model θ_{base}
Output: Synthetic dataset \mathcal{S}
Initialize \mathcal{S}
if θ_{base} *is from random or at early stage* **then**
 └ Prior-generate model pool \mathcal{M}
else if θ_{base} *is well-trained or at late stage* **then**
 └ Post-generation model pool \mathcal{M}
while *not converge* **do**
 └ Randomly select n models from \mathcal{M}
 └ Compute $\mathcal{L}(\mathcal{S}, \mathcal{T})$ as Eq. 7
 └ Back-propagate and update \mathcal{S}
Ensemble generate soft label via \mathcal{M} , $Y_s = \frac{1}{|\mathcal{M}|} \sum_{\theta \in \mathcal{M}} h(\mathcal{A}(X_s); \theta)$
 ▷ $h(\cdot; \theta)$ represents the model with parameter θ , \mathcal{A} is the function of data augmentation.
return \mathcal{S}

4 Experiments

In this section, we validate the efficacy of our proposed method, Teddy, via extensive experiments. Note that Teddy is designed for large-scale dataset distillation; we focus on two more challenging datasets, Tiny-ImageNet and full-size ImageNet. We also compare Teddy on the small datasets like CIFAR10 with fair comparison, showing its comparative performance (it is included in supplementary). We evaluate the cross-architecture generalization performance of our generated synthetic dataset and perform comprehensive ablation studies to demonstrate the effectiveness of the strategies adopted in our method.

4.1 Experiment Setting

Datasets and Networks To validate the efficacy and the efficiency of Teddy for large-scale datasets, we adopt two challenging datasets, 64×64 Tiny-ImageNet and 224×224 ImageNet-1K. While some previous methods have demonstrated effectiveness on downsized versions of ImageNet-1K or subsets, Teddy achieves impressive performance on the original, full-size ImageNet-1K. As for Tiny-ImageNet, following the instruction of MoCo (CIFAR) [12] and the prior work SRe²L [36], we adopt modified ResNet18 as the teacher model by replacing the first Conv layer with the 3×3 kernel, stride 1 Conv layer, and removing the first maxpool. For ImageNet-1K, we use ResNet18 [13], implemented by the official torchvision. Moreover, to show the generalization capabilities of our method, we evaluate the cross-architecture classification performance of ImageNet-1K with IPC 10 on several model architectures: ResNet50 [13], ResNet101 [13], DenseNet121 [14], MobileNetV2 [27], ShuffleNetV2 [21], and EfficientNetB0 [30].

Method	Tiny-ImageNet		ImageNet-1K		
	50	100	10	50	100
Random (Conv)	15.1 ± 0.3	24.3 ± 0.3	4.1 ± 0.1*	16.2 ± 0.8*	19.5 ± 0.5*
Random (ResNet18)	18.2 ± 0.2	25.0 ± 0.2	6.8 ± 0.1	32.0 ± 0.2	45.7 ± 0.1
DC [41]	11.2 ± 0.3	-	-	-	-
DSA [39]	25.3 ± 0.2	-	-	-	-
DM [40]	24.1 ± 0.3	29.4 ± 0.2	-	-	-
IDM [42]	27.7 ± 0.3	-	-	-	-
MTT [3]	28.2 ± 0.5	33.7 ± 0.6	-	-	-
FTD [10]	31.5 ± 0.3	34.5 ± 0.4	-	-	-
TESLA [5]	33.4 ± 0.5	34.7 ± 0.2	17.8 ± 1.3*	27.9 ± 1.2*	29.2 ± 1.0*
SRe ² L [36]	41.1 ± 0.4	49.7 ± 0.3	21.3 ± 0.6	46.8 ± 0.2	52.8 ± 0.3
Ours (post)	44.5 ± 0.2 (+ 3.4)	51.4 ± 0.2 (+ 1.7)	32.7 ± 0.2 (+ 11.4)	52.5 ± 0.1 (+ 5.7)	56.2 ± 0.2 (+ 3.4)
Ours (prior)	45.2 ± 0.1 (+ 4.1)	52.0 ± 0.2 (+ 2.3)	34.1 ± 0.1 (+ 12.8)	52.5 ± 0.1 (+ 5.7)	56.5 ± 0.1 (+ 3.7)

Table 1: Comparison with baseline methods. * indicates the evaluation results on downsampled ImageNet-1K dataset. Here, SRe²L and our proposed methods adopt the ResNet18 as the training and evaluation model, other methods adopt ConvNet.

Method	ResNet50	ResNet101	DenseNet121	MobileNetV2	ShuffleNetV2	EfficientNetB0
SRe ² L [36]	28.4 ± 0.1	30.9 ± 0.1	21.5 ± 0.5	10.2 ± 0.2	29.1 ± 0.1	16.1 ± 0.1
Ours (post)	37.9 ± 0.1 (+ 9.5)	40.0 ± 0.1 (+ 9.1)	33.0 ± 0.1 (+ 11.5)	20.5 ± 0.1 (+ 10.3)	40.0 ± 0.3 (+ 10.9)	27.3 ± 0.2 (+ 11.2)
Ours (prior)	39.0 ± 0.1 (+ 10.6)	40.3 ± 0.1 (+ 9.4)	34.3 ± 0.1 (+ 12.8)	23.4 ± 0.3 (+ 13.2)	38.5 ± 0.1 (+ 9.4)	29.2 ± 0.1 (+ 13.1)

Table 2: Evaluation results of cross-architecture generalization under the ImageNet-1K with IPC 10 setting. SRe²L and our methods use ResNet18 as the training model.

Implementation Details For prior model pool generation, we cache 9 checkpoints in the early stage of single teacher training trajectory to build the model pool for generating distilled data of ImageNet-1K, and 8 for Tiny-ImageNet. During the data generation phase, a subset of 3 models will be randomly selected to generate the i^{th} image for all classes, alleviating the mode collapse problem. As for the pruning-based model pool, we perform pruning on the off-the-shelf pre-trained ResNet18 provided by PyTorch, with the Top-1 accuracy of 69.76%, and finetune for very limited epochs, *e.g.*, 0-2 epochs. The GFLOPs of the target pruned model is 1.2G, and the number of parameters is 7.72M. Here, we save 10 models for generating distilled data of ImageNet-1K, and 9 for Tiny-ImageNet. We randomly select 4 models from the model pool to generate the synthetic dataset for ImageNet-1K and 5 models for Tiny-ImageNet. Moreover, after generation, the augmented synthetic data will input the model pool again to get soft labels. For more details, please refer to the supplementary materials.

Evaluation and Baselines Following prior works [33, 40, 41], we evaluate the distilled datasets by training several randomly initialized models from scratch and report the mean and standard deviation of their accuracy on the corresponding real test set. We compare our proposed methods with series methods, including DC [41], DSA [40], DM [40], IDM [42], MTT [3], FTD [10], TESLA [5] and SRe²L [36]. Here, we present the results derived from their original paper or DD benchmarks [4] if available for selected baseline methods. However, as most existing methods are challenging to apply to the full-sized ImageNet-1K, we also report results on downsized ImageNet-1K. Additionally, the methods adopted to

compare in the baseline all employ ConvNet, and SRe²L [36] and our method utilize ResNet18 as the training and evaluation model.

4.2 Results on Baselines

As demonstrated in Table 1, our proposed method, Teddy, significantly surpasses all the baselines for both the Tiny-ImageNet and ImageNet-1K datasets with all IPC settings. Furthermore, comparing the results across the two datasets reveals that our method exhibits a more pronounced improvement on complex datasets. In particular, in the case of ImageNet-1K with IPC 10, our evaluation results outperform the previous state-of-the-art methods by a large margin up to 12.8%. Moreover, our method achieves the same performance with the setting of IPC 50 (52.5%) as the previous SOTA method in the case of IPC 100 (52.8%). Additionally, for DD, enhancing the performance of the larger synthetic dataset is particularly challenging. This is because the existing DD methods focus mainly on key dataset information, ignoring long-tail information. With increased IPCs, there is less room to enhance as compressing these details is hard. However, as shown in Table 1, our method considerably improves the performance for large IPCs (5.7% and 3.7% for IPC 50 and 100). For the post-generation, although we applied pruning to the teacher model, resulting in a reduction of statistic information space in its batch normalization layers, the results in Table 1 indicate that the impact is not substantial. We can still achieve comparable performance.

4.3 Results on Cross-Architecture Generalization

The generalization capacity is vital for practically applying dataset distillation. Here, we evaluate the generalization performance of the distilled dataset of the ImageNet-1K IPC 10 setting. In selecting model architecture, we adopt the ResNet50, ResNet101, DenseNet121, MobileNetV2, ShuffleNetV2, and EfficientNetB0. All results are reported in Table 2.

The results show that our proposed method, Teddy, surpasses all previous state-of-the-art methods by a significant margin. When the evaluation models have larger and deeper architectures, there is a significant and proportional performance improvement, comparing 34.1%, 39.0%, and 40.3% for ResNet18, 50, and 101, with enhancements of 12.8%, 10.6% and 9.4%. For structurally different networks, our method demonstrates strong generalization capabilities. For instance, on DenseNet121, MobileNetV2, ShuffleNetV2, and EfficientNetB0, we achieved 34.3%, 23.4%, 40.0%, and 29.2%, respectively, surpassing the SOTA by 12.8%, 13.2%, 10.9%, and 13.1%. As shown in Table 2, the evaluation results of the post-generation method indicate a strong generalization capability, even with the architecture heterogeneity introduced by network pruning, surpassing the previous SOTA methods.

4.4 Ablation Study

To substantiate the efficacy of our method, we perform ablation studies explicitly focusing on the effect of the model pool and the weak teachers. The outcomes

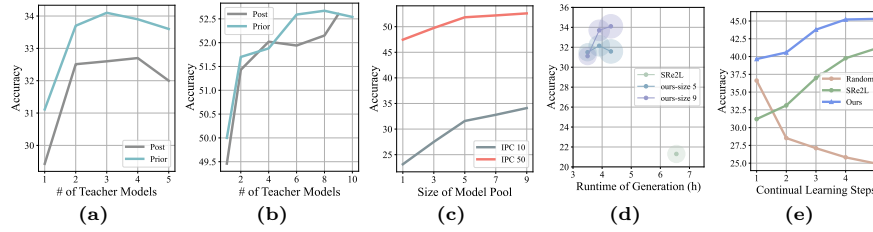


Fig. 3: (a) Ablation study on different number of models ensemble to generate synthetic data. (b) Ablation study on different number of models ensemble to generate soft label. (c) Ablation study on size of the model pool under the setting of ImageNet-1K IPC 10 and IPC 50 with the prior-generation strategy. (d) The time and memory requirement of our method compared with the previous SOTA. Here the size of the points represents the peak GPU memory, and the three points, from left to right, report the evaluation results of 1, 2, and 3 teacher models ensemble utilized in generating the synthetic data. (e) Continual learning on Tiny-ImageNet IPC 50 with 5-step incremental protocol.

underscore the effectiveness of the employed strategies, leading to a substantial enhancement in the performance of our proposed method.

Effect of Model Pool We utilize the model pool of weak teachers as a substitute for students, reducing computational expenses. To analyze effectiveness, we conduct separate experiments for data generation and relabel phases.

For the data generation part, we conduct experiments on the setting of ImageNet-1K with IPC 10, and the setting of the model pool is the same as the main table. For data generation, 1, 2, 3, 4, or 5 models will be randomly ensemble selected. The soft labels are generated by the whole model pool. The results in Fig. 3a reveal that the performance is better with multiple models than with a single model. The accuracy drops in Fig. 3b can be attributed to the trade-off between trying fit numerous models comprehensively or concentrating on fitting fewer ones. Also, our method exhibits robustness in terms of the number of models selected for generation.

As for the relabel part, we conduct experiments on the setting of ImageNet-1K with IPC 50, and the setting of the model pool is the same as the main table. For data generation, 3 models are randomly selected from the model pool. Those generated data will be ensemble relabeled by 1, 2, 4, 6, 8, or whole models of the model pool. The evaluation results are shown in Fig. 3b. We observe a significant enhancement in performance with ensemble relabeling, but also robust for the number of models selected for relabeling.

Effect of Weak Teacher We conduct ablation studies on the stage of weak teachers adopted in our method under the setting of ImageNet-1K with IPC 10, IPC 50, and IPC 100. The results are shown in Fig. 2c. Here, we generate the model pool with weak teachers in different stages. We cache 9 models for each trajectory segment with step 5. For data generation, we randomly select 3 out

of the model pool and use the whole models in the model pool to relabel the synthetic data. From the results, it is evident that the performance of teacher models significantly impacts the distilled data. For smaller IPCs, adopting the weak teacher models of the early stage is necessary.

Size of the Model Pool We also perform an ablation study on the size of the model pool, and the results are shown in Fig. 3c. The experiments are under the setting of ImageNet-1K IPC 10 and IPC 50. For one experiment setting, we cache the teacher models from the same trajectory segments; the beginning and ending points are the same as the main table but with different step lengths. Here, we randomly select 3 models from the model pool to generate data, and the whole model pool is adopted to generate the soft label. From the results, we observe that the size of the model pool does not significantly impact the performance of the generated synthetic data when the size is larger than 5. In this setting, a smaller size of the model pool may lead to an issue with mode collapse and a subsequent decline in performance.

4.5 Efficiency Evaluation

In this section, we evaluate the efficiency of our proposed method. The experiments are under the setting of ImageNet-1K IPC 10, utilizing 8 RTX 4090 GPUs. Here, the cached teachers are at stages 1 to 41. We report the total run time of model pre-training and data synthesizing, and the peak GPU memory of the data synthesizing process. The evaluation results are shown in Fig. 3d. From the results, due to the fact that we only need to train teacher models for a limited number of epochs (which is the major time overhead), the total time required for our method is significantly less than that of SRe²L. Noticing that when utilizing a single teacher model, the required GPU memory is the same as SRe²L, and the performance of the distilled data is notably superior to SRe²L.

4.6 Continual Learning

Continual learning (CL) [6, 25, 32] focuses on learning tasks sequentially while prior task data is unavailable. In order to prevent catastrophic forgetting, one typical strategy is using a small buffer to retain critical information from past tasks. In this case, the capacity of dataset distillation to retain the information of the original datasets within a confined storage space benefits practical applications in continual learning.

Here, we follow the previous work [40] employing the GDumb [23] as the base method. This method stores and combines past and new training data to train a model from scratch. We evaluate our proposed method on Tiny-ImageNet with IPC 50 following the 5-step class-incremental protocol. The results are shown in Fig. 3e. Our method significantly outperforms the baseline methods and demonstrates an accuracy trend that differs from random selection, gradually increasing with the addition of categories. This trend is attributed to the soft label strategy, which allows for incorporating more cross-class knowledge.

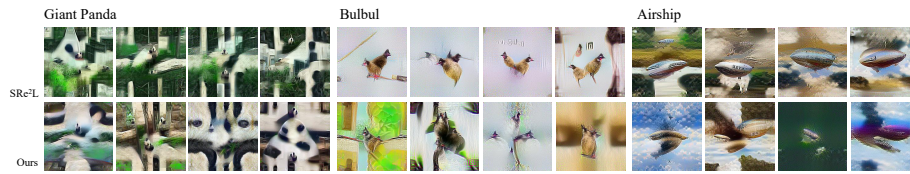


Fig. 4: Visualization of synthetic data generated by SRe²L and our method. The first row is generated by SRe²L, and the second row is generated by ours with the prior-generation strategy. Here, we choose three classes from ImageNet-1K: Giant Panda, Bulbul, and Airship.

4.7 Visualization

In this section, we present the visualization of ImageNet-1K synthetic datasets with IPC 50 setting generated by our method and SRe²L [36]. Fig. 4 shows that our method generates high quality, more distinctive features and a more comprehensive range of variations data, showing more perspective of the original dataset. In comparison, images generated from SRe²L, while realistic, face the problem of mode collapse, especially for large IPCs.

5 Conclusion

In this paper, we introduce Teddy, an efficient large-scale dataset distillation method that can handle the full-size ImageNet-1K. Back up with theoretical analysis, we propose a memory-efficient approximation derived from Taylor expansion to decouple the bi-level optimization of the original DD into a first-order one. To further improve the time efficiency, instead of training a novel model for each synthetic data update, we propose two efficient model pool generation strategies, the prior and post-generation, corresponding to two practical cases. Both strategies can generate the model pool from one single base model. Lastly, we adopt ensemble soft relabeling to improve the generalizability of the distilled data. Additionally, to the best of our knowledge, we are the first to build a unified theoretical framework of the DD, linking the existing mainstream DD methods. Various experiments and ablation studies show the effectiveness of our proposed method, outperforming the prior state-of-the-art DD methods by a significant margin.

Acknowledgements

This project is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award Number: MOE-T2EP20122-0006), and the National Research Foundation, Singapore, under its AI Singapore Programme (AISG Award No: AISG2-RP-2021-023).

References

1. Bohdal, O., Yang, Y., Hospedales, T.: Flexible dataset distillation: Learn labels instead of images. arXiv preprint arXiv:2006.08572 (2020)
2. Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y.T., Li, Y., Lundberg, S., et al.: Sparks of artificial general intelligence: Early experiments with gpt-4. arXiv preprint arXiv:2303.12712 (2023)
3. Cazenavette, G., Wang, T., Torralba, A., Efros, A.A., Zhu, J.Y.: Dataset distillation by matching training trajectories. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4750–4759 (2022)
4. Cui, J., Wang, R., Si, S., Hsieh, C.J.: Dc-bench: Dataset condensation benchmark. Advances in Neural Information Processing Systems **35**, 810–822 (2022)
5. Cui, J., Wang, R., Si, S., Hsieh, C.J.: Scaling up dataset distillation to imagenet-1k with constant memory. In: International Conference on Machine Learning. pp. 6565–6590. PMLR (2023)
6. De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. IEEE transactions on pattern analysis and machine intelligence **44**(7), 3366–3385 (2021)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
8. Deng, Z., Russakovsky, O.: Remember the past: Distilling datasets into addressable memories for neural networks. Advances in Neural Information Processing Systems **35**, 34391–34404 (2022)
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
10. Du, J., Jiang, Y., Tan, V.Y., Zhou, J.T., Li, H.: Minimizing the accumulated trajectory error to improve dataset distillation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3749–3758 (2023)
11. Fang, G., Ma, X., Song, M., Mi, M.B., Wang, X.: Depgraph: Towards any structural pruning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16091–16101 (2023)
12. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9729–9738 (2020)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
14. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
15. Kim, J.H., Kim, J., Oh, S.J., Yun, S., Song, H., Jeong, J., Ha, J.W., Song, H.O.: Dataset condensation via efficient synthetic-data parameterization. In: International Conference on Machine Learning. pp. 11102–11118. PMLR (2022)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)

17. Lee, H.B., Lee, D.B., Hwang, S.J.: Dataset condensation with latent space knowledge factorization and sharing. arXiv preprint arXiv:2208.10494 (2022)
18. Liu, S., Wang, K., Yang, X., Ye, J., Wang, X.: Dataset distillation via factorization. *Advances in Neural Information Processing Systems* **35**, 1100–1113 (2022)
19. Liu, S., Wang, X.: Mgdd: A meta generator for fast dataset distillation. *Advances in Neural Information Processing Systems* **36** (2024)
20. Liu, S., Ye, J., Yu, R., Wang, X.: Slimmable dataset condensation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 3759–3768 (2023)
21. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: *Proceedings of the European conference on computer vision (ECCV)*. pp. 116–131 (2018)
22. Nguyen, T., Chen, Z., Lee, J.: Dataset meta-learning from kernel ridge-regression. arXiv preprint arXiv:2011.00050 (2020)
23. Prabhu, A., Torr, P.H., Dokania, P.K.: Gdumb: A simple approach that questions our progress in continual learning. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II* 16. pp. 524–540. Springer (2020)
24. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125 **1**(2), 3 (2022)
25. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. pp. 2001–2010 (2017)
26. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 10684–10695 (2022)
27. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 4510–4520 (2018)
28. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
29. Sucholutsky, I., Schonlau, M.: Soft-label dataset distillation and text dataset distillation. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. pp. 1–8. IEEE (2021)
30. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: *International conference on machine learning*. pp. 6105–6114. PMLR (2019)
31. Wang, K., Zhao, B., Peng, X., Zhu, Z., Yang, S., Wang, S., Huang, G., Bilen, H., Wang, X., You, Y.: Cafe: Learning to condense dataset by aligning features. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 12196–12205 (2022)
32. Wang, L., Zhang, X., Su, H., Zhu, J.: A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024)
33. Wang, T., Zhu, J.Y., Torralba, A., Efros, A.A.: Dataset distillation. arXiv preprint arXiv:1811.10959 (2018)
34. Ye, J., Yu, R., Liu, S., Wang, X.: Distilled datamodel with reverse gradient matching. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 11954–11963 (2024)

35. Ye, J., Yu, R., Liu, S., Wang, X.: Mutual-modality adversarial attack with semantic perturbation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 6657–6665 (2024)
36. Yin, Z., Xing, E., Shen, Z.: Squeeze, recover and relabel: Dataset condensation at imagenet scale from a new perspective. arXiv preprint arXiv:2306.13092 (2023)
37. Yu, R., Liu, S., Wang, X.: Dataset distillation: A comprehensive review. arXiv preprint arXiv:2301.07014 (2023)
38. Zhang, L., Zhang, J., Lei, B., Mukherjee, S., Pan, X., Zhao, B., Ding, C., Li, Y., Xu, D.: Accelerating dataset distillation via model augmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11950–11959 (2023)
39. Zhao, B., Bilen, H.: Dataset condensation with differentiable siamese augmentation. In: International Conference on Machine Learning. pp. 12674–12685. PMLR (2021)
40. Zhao, B., Bilen, H.: Dataset condensation with distribution matching. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 6514–6523 (2023)
41. Zhao, B., Mopuri, K.R., Bilen, H.: Dataset condensation with gradient matching. arXiv preprint arXiv:2006.05929 (2020)
42. Zhao, G., Li, G., Qin, Y., Yu, Y.: Improved distribution matching for dataset condensation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7856–7865 (2023)
43. Zhou, Y., Nezhadarya, E., Ba, J.: Dataset distillation using neural feature regression. *Advances in Neural Information Processing Systems* **35**, 9813–9827 (2022)